



Step by Step Guide for Building a simple JSF Application (Guess a Number) - V1.0





Sang Shin

sang.shin@sun.com

www.javapassion.com

**Java™ Technology Evangelist
Sun Microsystems, Inc.**

Disclaimer & Acknowledgments

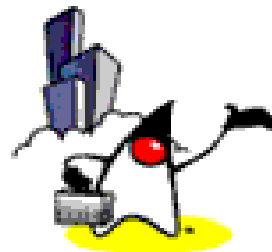
- Even though Sang Shin is a full-time employee of Sun Microsystems, the contents here is created as his own personal endeavor and thus does not reflect any official stance of Sun Microsystems.
- Sun Microsystems is not responsible for any inaccuracies in the contents.
- Acknowledgments:
 - Many slides and speaker notes are created from [JSF tutorial](#)
 - Source code examples are from sample codes that are shipped with JSF beta

Revision History

- 12/15/2003: version 1: created (Sang Shin)
- 12/21/2003: version 2: changed contents based on Beta version released on 12/20/2003 (Sang Shin)
- 01/03/2003: speaker notes are added (Sang Shin)
- 04/03/2004: updated to reflect V1.0 as part of J2EE 1.4 SDK
- 01/26/2007: small changes to reflect JSF 1.1
- Things to do
 - slides need polish



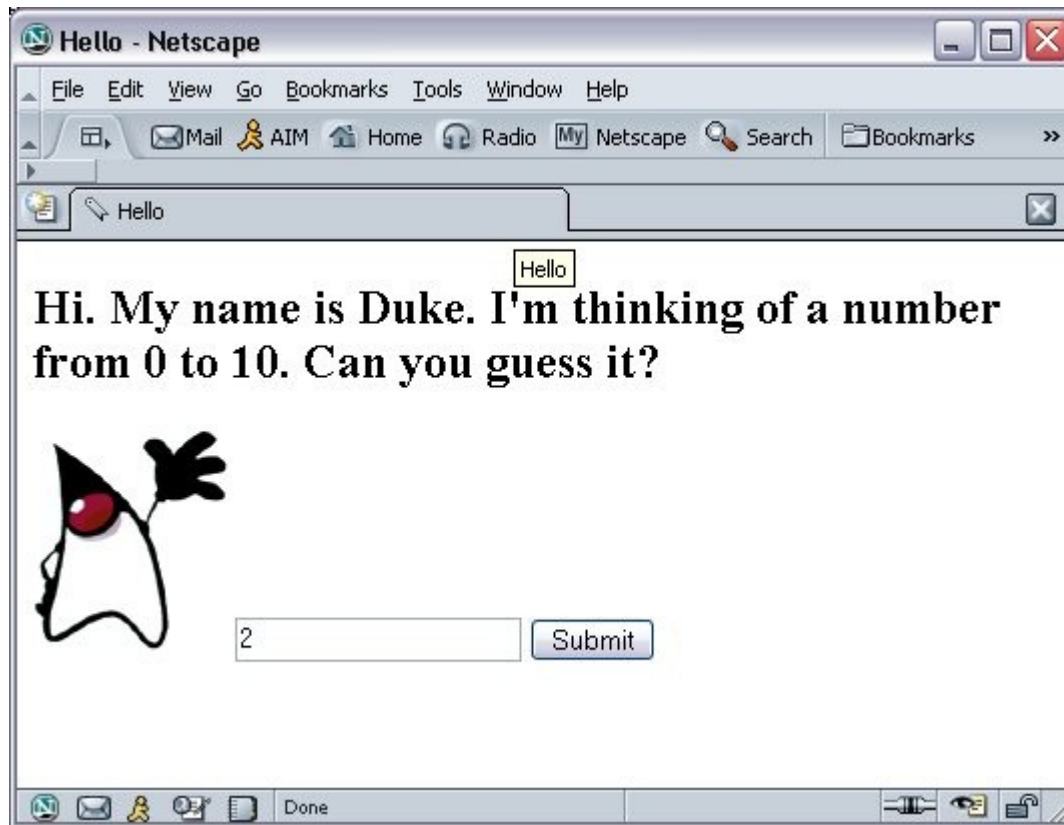
Guess a Number Sample Application



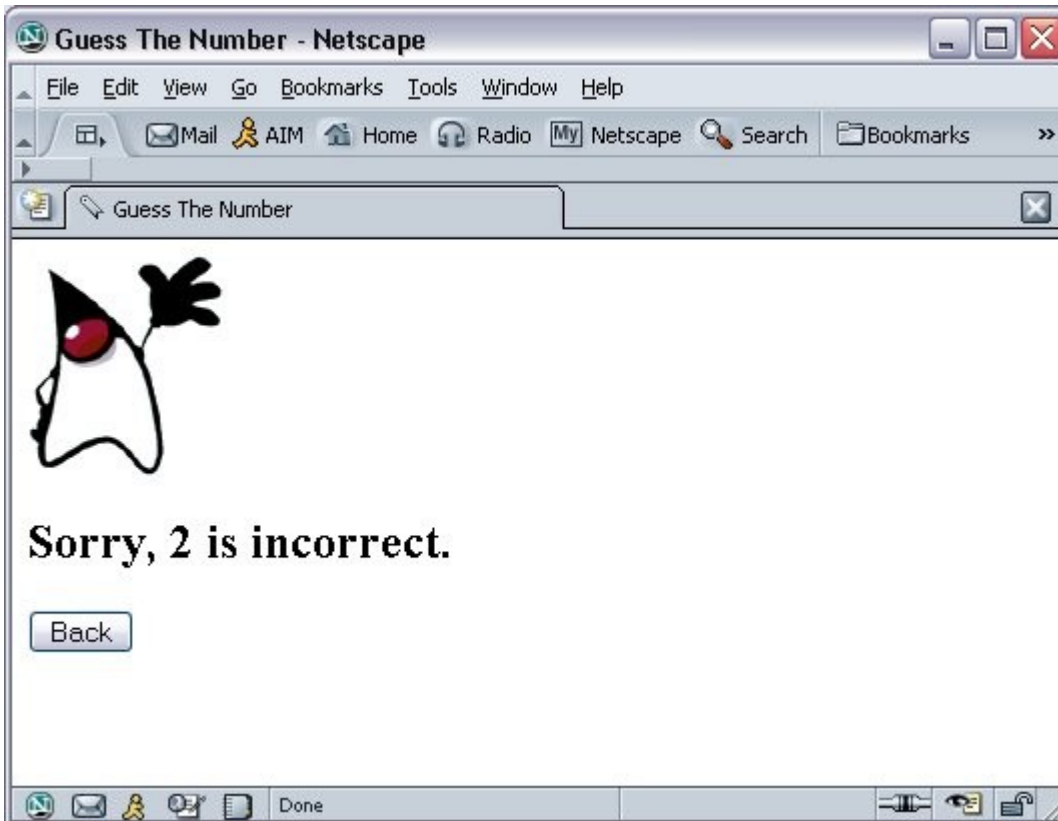
Sample JSP Application we are going to build

- `guessNumber` application that comes with JSF 1.0 as part of J2EE 1.4 SDK
- Guess a number between 0 and 10, inclusive
 - The response page tells you if you guessed correctly
- Input validation

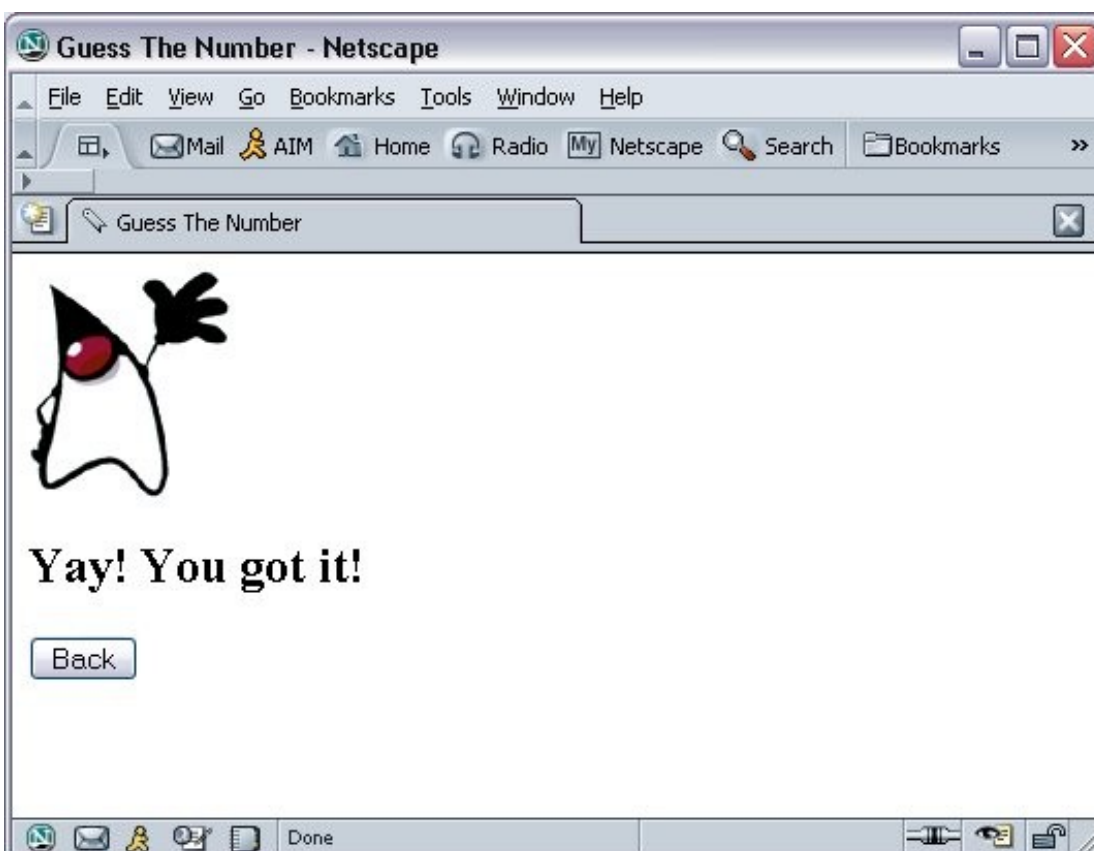
Starting Page



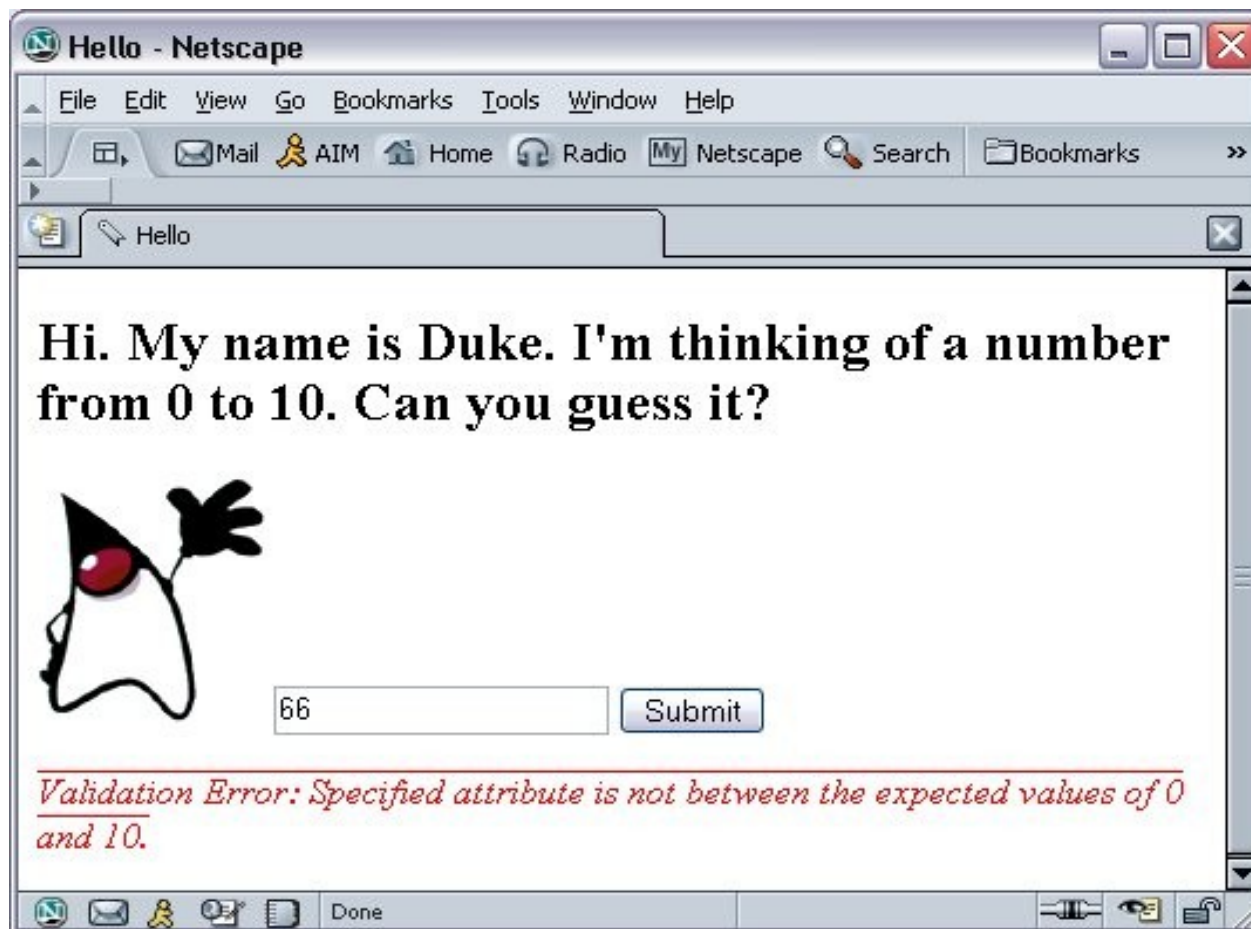
Failure Outcome Navigation



Success Outcome Navigation

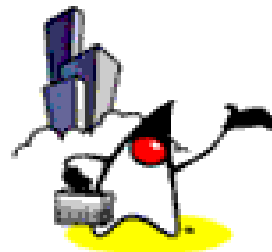


Input Validation Error





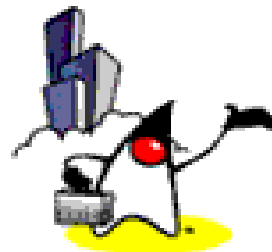
Steps to follow



Steps

1. Create development directory structure
2. Write web.xml
3. Write ant build script
4. JSF-specific steps
5. Build, deploy, and test the application

Step 1: Create Development Directory Structure



Development Directory Structure

- Same development directory structure for any typical Web application
 - We will use the source/build directory structure of sample Web applications that come with J2EE 1.4 SDK
- Ant build script should be written accordingly
 - Just use the build.xml script that comes with J2EE 1.4 SDK

*.jar files (1) - based in JSF (`<J2EE1.4_HOME>/lib`)

- jsf-api.jar
 - contains the javax.faces.* API classes
- jsf-impl.jar
 - contains the implementation classes of the JavaServer Faces standard implementation
- jstl.jar
 - required to use JSTL tags and referenced by JavaServer Faces standard implementation classes

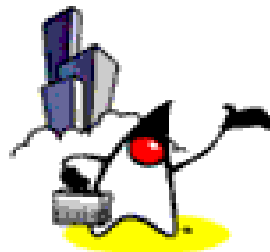
*.jar files (2)

- standard.jar
 - required to use JSTL tags and referenced by JavaServer Faces reference implementation classes
- commons-beanutils.jar
 - utilities for defining and accessing JavaBeans component properties
- commons-digester.jar
 - for processing XML documents

*.jar files (3)

- commons-collections.jar
 - extensions of the Java 2 SDK Collections Framework
- commons-logging.jar
 - a general purpose, flexible logging facility to allow developers to instrument their code with logging statements

Step 2: Write web.xml Deployment Descriptor



web.xml

- Same structure as any other Web application
 - `javax.faces.webapp.FacesServlet` is like any other servlet
 - Servlet definition and mapping of FacesServlet
- There are several JSF specific `<context-param>` elements

Example: web.xml

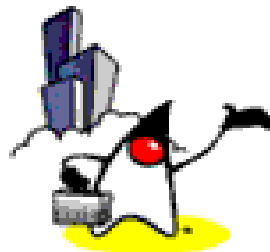
```
1 <web-app>
2   <display-name>JavaServer Faces Guess Number Sample Application
3   </display-name>
4   <description>
5     JavaServer Faces Guess Number Sample Application
6   </description>
7
8   <context-param>
9     <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
10    <param-value>client</param-value>
11  </context-param>
12
13  <context-param>
14    <param-name>javax.faces.application.CONFIG_FILES</param-name>
15    <param-value>/WEB-INF/faces-config.xml</param-value>
16  </context-param>
17
18  <context-param>
19    <param-name>com.sun.faces.validateXml</param-name>
20    <param-value>>true</param-value>
21  </context-param>
```

Example: web.xml

```
1
2     <listener>
3         <listener-class>com.sun.faces.config.ConfigListener</listener-class>
4     </listener>
5
6     <!-- Faces Servlet -->
7     <servlet>
8         <servlet-name>Faces Servlet</servlet-name>
9         <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
10        <load-on-startup> 1 </load-on-startup>
11    </servlet>
12
13
14    <!-- Faces Servlet Mapping -->
15    <servlet-mapping>
16        <servlet-name>Faces Servlet</servlet-name>
17        <url-pattern>/guess/*</url-pattern>
18    </servlet-mapping>
19
20 </web-app>
```



Step 3: Write Ant Build Script

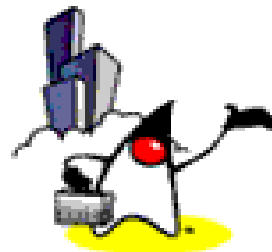


Ant build.xml

- Leverage the build.xml file of JSF sample apps or other Web application



Step 4: JSF-Specific Development Steps



“JSF-Specific” Steps for Developing JSF Application

1. Create the Pages

- Using the UI component and core tags

2. Define Page Navigation

- In the application configuration file

3. Develop the backing beans (Model objects)

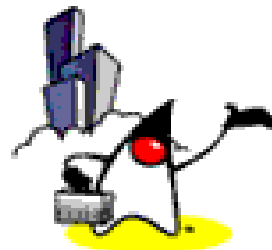
- Model objects hold the data
- Validator, convertor, event handler, navigation logic

4. Add managed bean declarations

- To the application configuration file



JSF Step1: Creating Pages



```
<HTML>  
  <HEAD> <title>Hello</title> </HEAD>  
  <%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>  
  <%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>  
  <body bgcolor="white">
```

```
<f:view>
```

```
<h:form id="helloForm" >
```

```
<h2>Hi. My name is Duke. I'm thinking of a number from
```

```
<h:outputText value="#{UserNumberBean.minimum}"/> to
```

```
<h:outputText value="#{UserNumberBean.maximum}"/>.
```

```
Can you guess it?
```

```
</h2>
```

```
<h:graphicImage id="waveImg" url="/wave.med.gif" />
```

```
<h:inputText id="userNo" value="#{UserNumberBean.userNumber}"  
  validator="#{UserNumberBean.validate}"/>
```

```
<h:commandButton id="submit" action="success" value="Submit" />
```

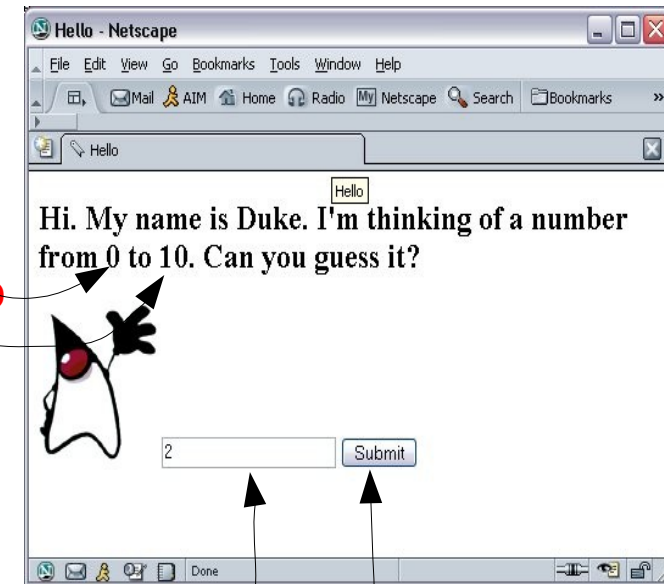
```
<p>
```

```
<h:message style="color: red; font-family: 'New Century Schoolbook', serif;  
  font-style: oblique; text-decoration: overline" id="errors1" for="userNo"/>
```

```
</h:form>
```

```
</f:view>
```

```
</HTML>
```



greeting.jsp (2) in (Input Validation Error)

```
<f:view>
```

```
<h:form id="helloForm" >
```

```
<h2>Hi. My name is Duke. I'm thinking of a number from
```

```
<h:outputText value="#{UserNumberBean.minimum}"/> to
```

```
<h:outputText value="#{UserNumberBean.maximum}"/>.
```

```
Can you guess it?
```

```
</h2>
```

```
<h:graphic_image id="waveImg" url="/wave.med.gif" />
```

```
<h:inputText id="userNo" value="#{UserNumberBean.userNumber}"  
    validator="#{UserNumberBean.validate}"/>
```

```
<h:commandButton id="submit" action="success" value="Submit" />
```

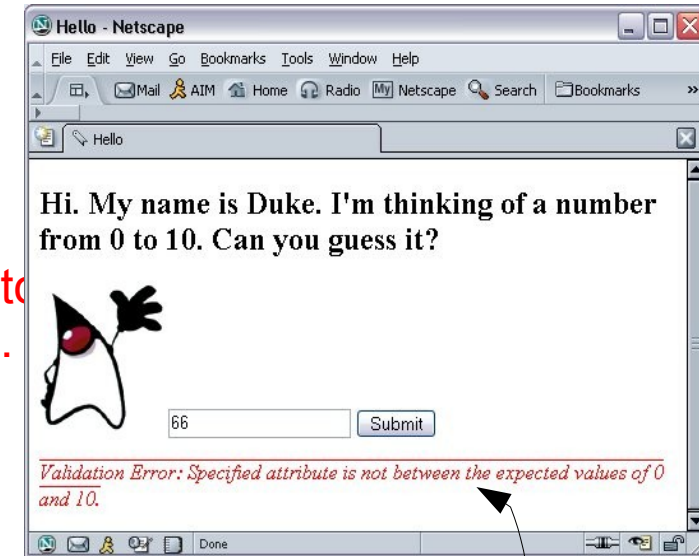
```
<p>
```

```
<h:message style="color: red; font-family: 'New Century Schoolbook', serif;  
    font-style: oblique; text-decoration: overline" id="errors1" for="userNo"/>
```

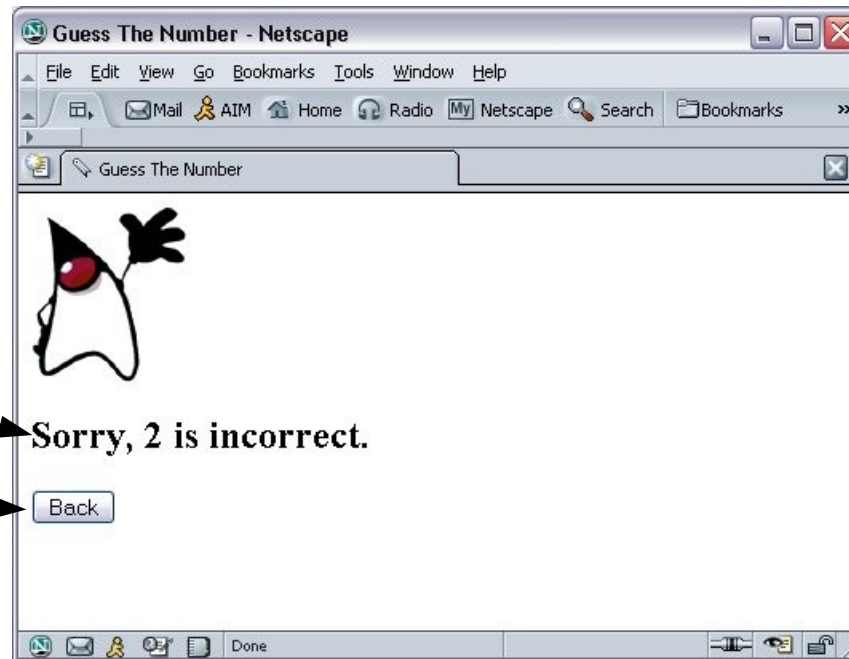
```
</h:form>
```

```
</f:view>
```

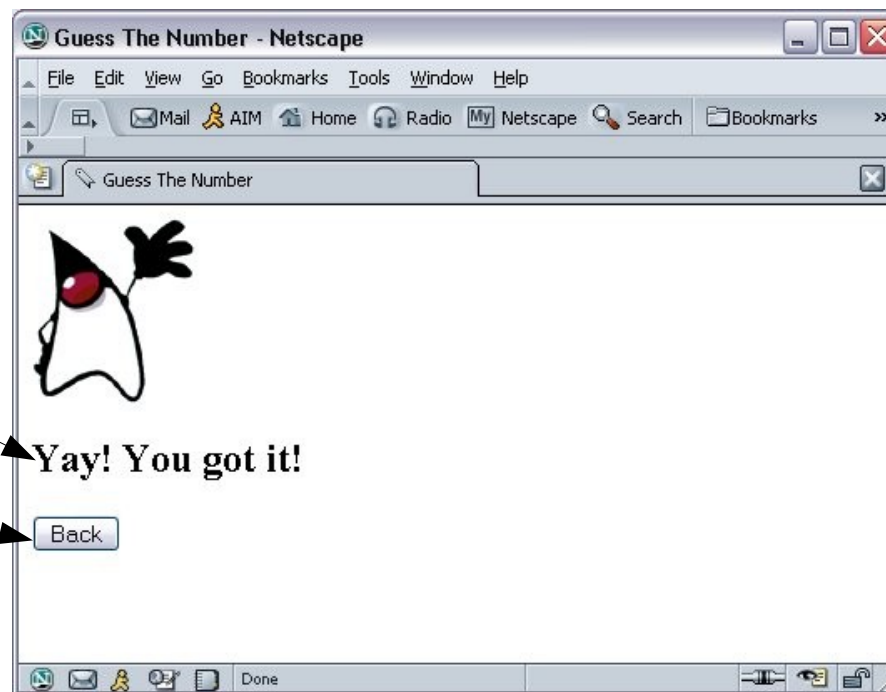
```
</HTML>
```



```
<HTML>
<HEAD> <title>Guess The Number</title> </HEAD>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<body bgcolor="white">
<f:view>
<h:form id="responseForm" >
    <h:graphicImage id="waveImg" url="/wave.med.gif" />
<h2><h:outputText id="result"
    value="#{UserNumberBean.response}"/></h2>
<h:commandButton id="back" value="Back" action="success"/><p>
</h:form>
</f:view>
</HTML>
```



```
<HTML>
  <HEAD> <title>Guess The Number</title> </HEAD>
  <%@ taglib uri="http://java.sun.com/jsp/html" prefix="h" %>
  <%@ taglib uri="http://java.sun.com/jsp/core" prefix="f" %>
  <body bgcolor="white">
    <f:view>
      <h:form id="responseForm" >
        <h:graphicImage id="waveImg" url="/wave.med.gif" />
        <h2><h:outputText id="result"
          value="#{UserNumberBean.response}"/></h2>
        <h:commandButton id="back" value="Back" action="success"/><p>
    </h:form>
  </f:view>
</HTML>
```



Creating Pages

- Page author's responsibility
- Involves
 - Laying out UI components on the pages
 - `<h:outputText>`, `<h:inputText>`, `<h:commandButton>`
 - Mapping the components to backing beans (model object data)
 - `#{UserNumberBean.minimum}`
 - Add other JSF features (either as tags or attributes)
 - `validator=#{UserNumberBean.validate}"`

greeting.jsp: `<h:form ...>` tag

- Represents an input form, which allows the user to input some data and submit it to the server, usually by clicking a button
- **UIInput** and **UIOutput** components are nested inside
 - `<h:inputText>`
 - `<h:commandButton>`
 - `<h:outputText>`

greeting.jsp: <h:outputText> tag

```
<h:outputText value="#{UserNumberBean.minimum}"/>  
<h:outputText value="#{UserNumberBean.maximum}"/>
```

- value="#{X.Y}" attribute
 - X matches the name defined by the **managed-bean-name** element corresponding to the proper managed-bean declaration from the application configuration file
 - Y matches name defined by the **property-name** element corresponding to the proper **managed-property** element

Relationship between JSP page and Backing Bean Declaration in App. Conf. File

- greeting.jsp

```
<h:outputText value="#{UserNumberBean.minimum}"/>
```

```
<h:outputText value="#{UserNumberBean.maximum}"/>
```

- Application configuration file (faces-config.xml)

```
<managed-bean>
```

```
  <description>
```

The "backing file" bean that backs up the guessNumber webapp

```
  </description>
```

```
  <managed-bean-name>UserNumberBean</managed-bean-name>
```

```
  <managed-bean-class>guessNumber.UserNumberBean</managed-bean-class>
```

```
  <managed-bean-scope>session</managed-bean-scope>
```

```
  <managed-property>
```

```
    <property-name>minimum</property-name>
```

```
    <property-class>java.lang.Long</property-class>
```

```
    <value>0</value>
```

```
  </managed-property>
```

```
  <managed-property>
```

```
    <property-name>maximum</property-name>
```

```
    <property-class>java.lang.Long</property-class>
```

```
    <value>10</value>
```

```
  </managed-property>
```

```
</managed-bean>
```

Relationship between JSP page and Backing Bean Class

- greeting.jsp

```
<h:outputText value="#{UserNumberBean.minimum}"/>  
<h:outputText value="#{UserNumberBean.maximum}"/>
```

- UserNumberBean.java

```
public class UserNumberBean {  
    private int maximum = 0;  
    public int getMaximum() {  
        return (this.maximum);  
    }  
    public void setMaximum(int maximum) {  
        this.maximum = maximum; ...  
    }  
  
    private int minimum = 0;  
    public int getMinimum() {  
        return (this.minimum);  
    }  
    public void setMinimum(int minimum) {  
        this.minimum = minimum; ...  
    }  
}
```

greeting.jsp: <h:inputText> tag

```
<h:inputText id="userNo" value="#{UserNumberBean.userNumber}"  
            validator="#{UserNumberBean.validate}"/>
```

- `value="#{UserNumberBean.userNumber}"`
 - The `value` attribute refers to the `userNumber` property on the `UserNumberBean` bean
 - After the user submits the form, the value of the `userNumber` property in `UserNumberBean` will be set to the text entered in the field corresponding to this tag

Relationship between JSP page and Backing Bean Class

- greeting.jsp

```
<h:inputText id="userNo" value="#{UserNumberBean.userNumber}"  
            validator="#{UserNumberBean.validate}"/>
```

- UserNumberBean.java

```
public class UserNumberBean {  
  
    Integer userNumber = null;  
    public void setUserNumber(Integer user_number) {  
        userNumber = user_number;  
        System.out.println("Set userNumber " + userNumber);  
    }  
  
    public Integer getUserNumber() {  
        System.out.println("get userNumber " + userNumber);  
        return userNumber;  
    }  
    ...  
}
```

greeting.jsp: <h:inputText> tag

```
<h:inputText id="userNo" value="#{UserNumberBean.userNumber}"  
  validator="#{UserNumberBean.validate}"/>
```

- `validator="#{UserNumberBean.validate}"`
 - Is a JSF EL expression pointing to a backing-bean method that performs validation on the component's data.

Relationship between JSP page and Backing Bean Class (page 1)

- greeting.jsp

```
<h:inputText id="userNo" value="#{UserNumberBean.userNumber}"  
            validator="#{UserNumberBean.validate}"/>
```

- UserNumberBean.java

```
public class UserNumberBean {
```

```
    ...
```

```
    public void validate(FacesContext context, UIInput component, Object value) {  
        int v = ((Integer) value).intValue();  
        if (v < minimum || v > maximum) {  
            FacesMessage message = new FacesMessage();  
            message.setDetail(detailMessage);  
            message.setSummary(summaryMessage);  
            message.setSeverity(FacesMessage.SEVERITY_ERROR);  
            throw new ValidatorException(message);  
        }  
    }  
}
```

greeting.jsp: <h:commandButton ...> tag

```
<h:commandButton id="submit" action="success"  
value="Submit" />
```

- Represents the button used to submit the data entered in the text field
- **action** attribute specifies an outcome that helps the navigation mechanism to decide which page to display next

greeting.jsp: <h:message ...> tag

```
<h:message style="color: red; font-family: 'New Century Schoolbook', serif; font-style: oblique; text-decoration: overline" id="errors1" for="userNo"/>
```

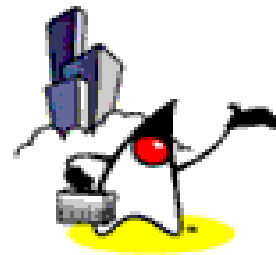
- **for** attribute refers to the ID of the component that generated the error messages
- **<h:message>** tag will display the error messages wherever the message tag appears in the page.
- **style** attribute allows you to specify the style of the text of the message. In the example, the text will be red, New Century Schoolbook, serif font family, oblique style, and a line will appear over the text

```
<f:view>
  <h:form id="helloForm" >
    <h2>Hi. My name is Duke. I'm thinking of a number from
      <h:outputText value="#{UserNumberBean.minimum}"/> to
      <h:outputText value="#{UserNumberBean.maximum}"/>.
      Can you guess it?
    </h2>

    <h:graphicImage id="waveImg" url="/wave.med.gif" />
    <h:inputText id="userNo" value="#{UserNumberBean.userNumber}"
      validator="#{UserNumberBean.validate}"/>
    <h:commandButton id="submit" action="success" value="Submit" />
    <p>
    <h:message style="color: red; font-family: 'New Century Schoolbook', serif;
      font-style: oblique; text-decoration: overline" id="errors1" for="userNo"/>

    </h:form>
  </f:view>
```

JSF Step 2: Define Page Navigation



Define Page Navigation

- Application developer responsibility
 - Navigation rules are defined in the application configuration file
- Navigation rules
 - Determine which page to go to after the user clicks a button or a hyperlink

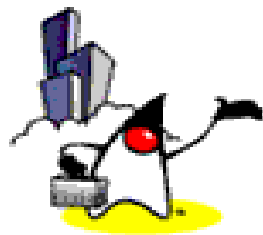
Navigation Rule 1 for guessNumber Example

```
<navigation-rule>
  <description>
    The decision rule used by the NavigationHandler to
    determine which view must be displayed after the
    current view, greeting.jsp is processed.
  </description>
  <from-view-id>/greeting.jsp</from-view-id>
  <navigation-case>
    <description>
      Indicates to the NavigationHandler that the response.jsp
      view must be displayed if the Action referenced by a
      UICommand component on the greeting.jsp view returns
      the outcome "success".
    </description>
    <from-outcome>success</from-outcome>
    <to-view-id>/response.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
```

Navigation Rule 2 for guessNumber Example

```
<navigation-rule>
  <description>
    The decision rules used by the NavigationHandler to
    determine which view must be displayed after the
    current view, response.jsp is processed.
  </description>
  <from-view-id>/response.jsp</from-view-id>
  <navigation-case>
    <description>
      Indicates to the NavigationHandler that the greeting.jsp
      view must be displayed if the Action referenced by a
      UICommand component on the response.jsp view returns
      the outcome "success".
    </description>
    <from-outcome>success</from-outcome>
    <to-view-id>/greeting.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
```

JSF Step 3: Developing Backing Beans (Model objects)



Developing Model Object

- Model object holds data
 - It is a JavaBean
- Responsibility of application developer
 - Page author and the application developer need to work in tandem to make sure that the **UI component tags refer to the proper object properties**

Model object:UserNumberBean

```
package guessNumber;

import java.util.Random;

import javax.faces.component.UIInput;
import javax.faces.context.FacesContext;
import javax.faces.validator.Validator;
import javax.faces.validator.LongRangeValidator;

public class UserNumberBean {

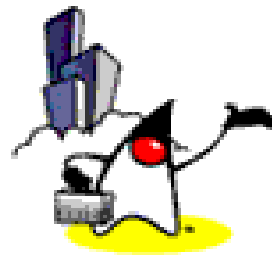
    Integer userNumber = null;
    Integer randomInt = null;
    String response = null;

    public UserNumberBean () {
        Random randomGR = new Random();
        randomInt = new Integer(randomGR.nextInt(10));
        System.out.println("Duke's number: "+randomInt);
    }
}
```

Properties of Model Object

- Property can be any of the basic primitive and reference types
 - Number types, String, int, double, and float
- JSF implementation automatically convert the data to the type specified by the model object property
- You can also apply a **converter** to a component to convert the components value to a type not supported by the component.

JSF Step 4: Adding Managed Bean Declarations



Adding Managed Bean Declarations to App. Conf. File

- JSF implementation processes Application Configuration File on application startup time and initializes the **UserNumberBean** and stores it in session scope if no instance exists
- Bean is then available for all pages in the application
- No need for `<jsp:useBean>` tag

UserNumberBean in faces-config.xml

```
<managed-bean>  
  <description>  
    The "backing file" bean that backs up the guessNumber webapp  
  </description>  
  <managed-bean-name>UserNumberBean</managed-bean-name>  
  <managed-bean-class>guessNumber.UserNumberBean</managed-bean-  
  class>  
  <managed-bean-scope>session</managed-bean-scope>  
  
  <managed-property>  
    <property-name>minimum</property-name>  
    <property-class>java.lang.Long</property-class>  
    <value>0</value>  
  </managed-property>  
  <managed-property>  
    <property-name>maximum</property-name>  
    <property-class>java.lang.Long</property-class>  
    <value>10</value>  
  </managed-property>  
</managed-bean>
```



**Live your life
with Passion!**

